Série 10 :
Programmation *C* - Arguments de la ligne de commande et précompilation

Buts

Le but de cette série d'exercices est de vous permettre de pratiquer les arguments de la ligne de commande et la précompilation.

Rappel

Avez-vous pris connaissance des **conseils relatifs à ces séries d'exercices** ?

Exercice 1 : QCM re-revisités (arguments de la ligne de commande, niveau 1)

On va prendre l'**exercice 2 de la série 8** pour en modifier le comportement de sorte à ce que le nom du fichier à lire puisse être :

- soit passé en argument dans la ligne de commande ;
- soit demandé avec la fonction `demander_fichier()` si aucun argument n'a été passé dans la ligne de commande.

Prenez garde à ce que le programme se comporte « bien » :

1. Si plus d'un argument sont fournis, il renvoie un message d'erreur (et une explication).
2. Si l'argument fourni n'est pas un nom de fichier lisible, il donne un message d'erreur.

Exemples d'interaction

Aucun argument: la fonction `demander_fichier` est utilisée pour obtenir un nom de fichier valide:

```
./questionnaire
Nom du fichier à lire : ../qcm.tx
ERREUR, je ne peux pas lire le fichier ../qcm.tx
../qcm.tx: No such file or directory
Nom du fichier à lire : rien
ERREUR, je ne peux pas lire le fichier rien
rien: No such file or directory
Nom du fichier à lire : encorerien
ERREUR, je ne peux pas lire le fichier encorerien
encorerien: No such file or directory
=> j'abandonne !

./questionnaire
Nom du fichier à lire : ../qcm.txt
-> OK, fichier ../qcm.txt ouvert pour lecture.
Combien de dents possède un éléphant adulte ?
    1- 32
    2- de 6 à 10
...
```

Un argument: on tente d'ouvrir le fichier:

```
./questionnaire ../qcm.txt
-> OK, fichier ../qcm.txt ouvert pour lecture.
Combien de dents possède un éléphant adulte ?
    1- 32
...

./questionnaire rien
ERREUR, je ne peux pas lire le fichier rien
rien: No such file or directory
argument invalide !
```

Plus d'un argument: le programme retourne une erreur suivie d'une explication.

```
./questionnaire ../qcm.txt rien
ERREUR: trop d'arguments !
Vous devez fournir *un* nom de fichier.
```

Note: Faites attention de ne pas dupliquer inutilement le code!

Exercice 2 : Retour sur les piles (niveau 1)

Reprennez un de vos programmes travaillant sur les piles (voir **exercice 2** ou **3** de la série 9) et ajoutez une directive de compilation `DEBUG` de sorte que l'état de la pile soit, ou non, affiché après chaque opération sur la pile.

Je vous conseille pour cela d'utiliser une fonction `affiche_pile()`.

Exercice 3: Word Count (niveau 2, dernière partie: niveau 3)

Le but est ici d'en écrire un, similaire à la commande `wc` («Word Count», `man wc` pour ceux qui ne se connaissent pas, disponible aussi **ici**).

Version 1

Écrire un programme `cm.c` («cm» pour «Compteur de Mots» `;-)`) pouvant prendre 1 ou plusieurs noms de fichiers en argument et, pour chacun des fichiers, compte le nombre de blancs (espaces, tabulation ou retour à la ligne, etc... Utilisez pour cela la fonction `isspace` définie dans `ctype.h` (`man isspace` pour plus de détails)).

Note: pour lire un fichier caractère par caractère on utilisera la fonction `getc` (comme d'hab., `man getc` pour plus de détails)

Exemple d'utilisation à ce stade :

```
cm bidule.txt
bidule.txt : 12

cm machin.c truc.l bidule.txt
machin.c : 1203
truc.l : 542
bidule.txt : 12
```

Si l'un des fichiers n'est pas lisible, `cm` affichera un message d'erreur adéquat (utilisez par exemple `perror` : `man perror` pour en savoir plus) et la valeur de retour doit être égale à `1`.

Il faudra de plus que les messages d'erreur soient bien envoyés sur la sortie d'erreur (et non pas la sortie standard ! voir le dernier exemple ci-dessous).

Exemple d'interaction :

```
cm machin.c truc.l bidule.txt
machin.c : 1203
truc.l : 542
bidule.txt : 12
echo $?
0
chmod -r truc.l
cm machin.c truc.l bidule.txt
machin.c : 1203
truc.l : Permission denied
bidule.txt : 12
echo $?
1
rm truc.l
cm machin.c truc.l bidule.txt
machin.c : 1203
truc.l : No such file or directory
bidule.txt : 12
echo $?
1
sh
$ cm machin.c truc.l bidule.txt 2>/dev/null
machin.c : 1203
bidule.txt : 12
$ exit
```

Version 2

Modifiez maintenant votre programme pour que sans argument la commande `cm` compte le nombre de mots sur l'entrée standard.

Exemple :

```
cm < bidule.txt
1203
```

[Niveau 3] Pour continuer, vous pouver ajouter des options genre `-l` pour compter le nombre de lignes au lieu du nombre de caractères.

[Niveau 3] Et pour aller encore plus loin, vous pouvez aller voir le code de GNU `wc.c`:

- **ici en local (version 8.23 du 18 juillet 2014)** ;
- **ou ici dans leur git**.

  et chercher à tout comprendre, à comprendre chaque ligne...

Dernière mise à jour le 6 mai 2022
Last modified: Fri May 6, 2022